Graphic Pages

5 October 2006

Graphics telemetry pages

Graphics pages are telemetry pages that can display mnemonic values and expressions in text form or in objects like plots and gauges.

They are created with the ITOS tool makepage which can be run from a shell prompt or by clicking on the STOL "`prog`" button and selecting "`Display builder`".

They are saved in files with the "`.disp`" extension.

They can be displayed:

1. With the STOL PAGE directive.
2. By running selectpage from a shell prompt.
3. Via a web browser using select_page.cgi and the `ITOS.javadisplay.pages.IselectPage` applet.

Creating a graphic telemetry page
Modifying a page
Saving a page
Page builder menus
Global properties
Moving and deleting multiple objects
Viewing live data
Getting information at runtime

Page objects available:
Static text
Mnemonic
Expression
Bar gauge
Dial gauge
Command Button
Plot
Stripchart
Rectangle
RoundRectangle
Ellipse
Line
Arc
Label

Page object requirements contains information useful to developers of new page objects.

1 Creating a graphic telemetry page

Graphics pages are created with the ITOS tool makepage which can be run from a shell prompt or by clicking on the STOL `prog` button and selecting "Display builder".

During the first part of the page builder's initialization, it displays the message "`Loading page elements...`". When that small message window disappears, the page builder is ready for use.

Three windows are now displayed:

1. **Toolbox** contains the objects that can be added to the page.
2. **Properties** contains the property editor. It displays the properties for the object that has focus. When the page builder first starts, the top-level container has focus, so the global properties are displayed.
3. **ITOS page builder** contains the page being built.

Common Activities:

- Adding an object
- Deleting an object
- Moving objects
- Changing an object's properties
- Copying an object
- Saving a page

A striped border is drawn around the object that has focus. Change focus by clicking on a different object or off any object. The property editor allows you to modify the object that has focus. Entries in the `Edit` menu also apply to the object that has focus.

If you click off any object, the top-level container is given focus. The striped border is drawn around the whole page and the property editor displays global properties.

You can move and delete multiple objects.

Many errors are easier to detect and fix in the page builder if you display live data.

1.1 Adding an object to a page

To add an object to a page:

1. Click on the object's name in the **Toolbox** window. The cursor will change to a plus sign shape.
2. Move the cursor to the page window and click where you want the upper left corner of the object. (The page window is named "`ITOS Page Builder`" or the name of the file you are editing.)

3. The object will appear and the property editor will display its properties. You will have to edit those properties to make the object do what you want.
4. If you want to change the object's properties at a later time, click on the object to give it focus. That will display its property editor.

If you change your mind after the first step above, you can get rid of the plus sign cursor by clicking on the same object's name in the `Toolbox` window.

1.2 Deleting an object from a page

To delete an object from a page:

1. Give the object focus by clicking on it. A striped border is drawn around the object when it has focus.
2. Click on the `Delete` item in the `Edit` menu. That removes the object from the page.

It is also possible to delete multiple objects.

1.3 Moving objects

To move a single object:

1. Give it focus by clicking on it. A striped border is drawn around the object.
2. Put the cursor on the striped border. The cursor will change to a plus sign shape. If the cursor changes to an arrow and right angle, it is too close to a corner of the border. Move it toward the center of one of the sides.
3. Press a mouse button, drag the cursor to the desired location and release the mouse button.

A move can be influenced by the global settings. If `"Snap elements to grid?"` is set to true, the object's top left corner is put on the grid intersection closest to where you released the mouse button. That can be a help if you want objects to line up horizontally or vertically. The global settings can be used to set the grid's horizontal and vertical spacing.

You can also move several objects as a group.

1.4 Changing an object's properties

To change an object's properties:

1. Give the object focus by clicking on it. That brings up the object's property editor.
2. Click on the property you want to edit. You will have to click on the right hand half of the property's line. For example, don't click on "Background", but on the colored rectangle to the right.
3. Enter the desired value. For some properties, like colors and fonts, that may mean selecting from a custom editor.
   For other properties you will type the desired value. **Press return** after entering a value or the value will be ignored.

You may close the `Properties` window, to re-open it give a new object focus by left-clicking on it.

1.5 Copying an object

To copy an object:

1. Give the object focus by clicking on it.
2. Click on the `Copy` item in the `Edit` menu.
3. Click on the `Paste` item in the `Edit` menu.
4. Move the cursor onto the page. The cursor will have the same plus sign shape as when an object is added to the page.
5. Click where you want the top left corner of the object.

After selecting `Copy`, the object remains in the copy buffer until a different object is put there by another copy command. So you can do any number of paste operations without another copy.

The properties in the pasted object are initially the same as those in the copied object. That can be useful if you want a set of objects with properties different from the defaults. For example, a set of gauges with the same shape different from the default shape.

It is not possible to copy more than one object at a time. When more than one object is highlighted, the `Copy` item of the `Edit` menu is inactive.

1.6 Saving a Graphics Page

To save the current contents of the page builder in a file:

1. Select `"Save As..."` in the `File` menu. That brings up the `"Save"` dialog.
2. Specify the path and file name. You can do that in one step by typing the complete path in the text field labelled `"File Name"` or you can click on the other components in the `Save` dialog to browse through `Folders` and locate the desired `File`.
3. ITOS can display a graphics page only if it is saved in a file with the suffix `".disp"` and if the file name does not contain any upper case letters. A warning dialog will be displayed if those conditions are not met.

2 Modifying a Graphics Page

To modify a graphics page that was previously saved in a file:

1. Select "Open..." from the "File" menu. If this generates errors, see item 4 below.
2. Specify the path and file name. You can do that in one step by typing the complete path in the text field labelled "File Name" or you can click on the other components in the Open dialog to browse through Folders and locate the desired File.
3. The contents of the saved file will be displayed in the page builder. Any previous contents of the page builder are erased. A warning message will be displayed if the previous contents were not saved before the load operation.
4. Updates to ITOS may cause errors when attempting to Open a graphics page; if this occurs, the itos_convert script may need to be executed. Convert re-serializes unusable graphics pages into usable pages built with updated class files. See itos_convert for more details.

3 Page Builder Menus

3.1 File Menu

`New` Is used to open a new `ITOS Page Builder` window.

`Open...` Is used to load a file into the page builder for modification. Modifying a page gives more information.

`Save As...`
Is used to save the current contents of the page builder in a file. Saving a page gives more information.

`Print...` Is used to print the current contents of the page builder. A dialog is displayed which allows you to specify a printer or disk file.

`Close` Closes the `ITOS Page Builder` window, this menu item will quit the `ITOS Page Builder` application if there are no longer any other active `ITOS Page Builder` windows.

`Quit` Exits the `ITOS Page Buidler` application. The user will be prompted to confirm this action if there is more than one active `ITOS Page Builder` window.

3.2 Edit Menu

`Cut` Removes the in-focus object from the page and puts it into the paste buffer. This menu item is inactive when the top-level container has focus.

`Copy` Puts the in-focus object into the paste buffer so it can later be inserted with the `"Paste"` item. This item is inactive when the top-level container has focus or when more than one object has focus.

`Paste` Places an object from the paste buffer onto the page. This menu item is active only after either the `"Cut"` or `"Copy"` item has been selected. Copying gives more information.

`Delete` Permenantly removes the selected objects from the page, these objects may not be pasted later. Multiple objects gives more information.

3.3 Data Menu

`Start` Starts the ingestion of live data from the data point server. Viewing live data gives more information.

`Stop` Stops the ingestion of live data from the data point server. Viewing live data gives more information.

`Properties...`
Opens the `Data Source` properties dialog. Viewing live data gives more information.

3.4 Help Menu

`Mnemonics`
: This item brings up a Web browser instance and displays the table of contents for the mission database's mnemonics, commands, packet maps and subsystems.

`Page Builder`
: This item brings up an instance of a Web browser and displays this document.

The `Mnemonics` and `Page Builder` menu items rely on the environment variable $ITOS_URLBROWSER to determine which Web browser should be launched and the environment variable $ITOS_URLPREFIX to determine where the documentation is located.

4 Global Properties

Global properties are the properties displayed in the property editor when the top-level container has focus. That is, when the striped border surrounds the whole page.

The global properties include:

`BackgroundColor`
: Specifies the page's background color. Current and future text objects use this background color unless their backgrounds have been explicitly set.

`Font`
: Specifies the default font used in text objects added in the future. Changing this property does not change any objects already on the page.

`ForegroundColor`
: Specifies the default foreground color for text objects. Changing this property affects all current and future text objects except for those whose foreground color is set in their individual property editors.

`Grid color`
: Specifies the color used to display the grid.

`Snap elements to the grid?`
: When this property is set to true, all objects added to the page and all objects moved are placed so their top left corners are on a grid intersection point.
No objects are moved when this property is changed to true. Only objects added or moved in the future will be affected.
Note that this property is effective whether the grid is displayed or not. By default, the grid is not displayed and this property is true.

`Display a grid?`
: When this property is set to true, a rectangular grid is drawn on the page. This can help with lining objects up.

`Horizontal grid spacing`
: Specifies the number of pixels between vertical grid lines. Values smaller than 4 are ignored.

`Vertical grid spacing`
: Specifies the number of pixels between horizontal grid lines. Values smaller than 4 are ignored.

As with all property editors, to edit a property you give the property focus by clicking on the right half of the property's row. For example, to change the font, click on the font family name drop-down box and not on `"Font"`.

If a property is changed by typing text, press the return key after entering the text.

5 Moving and Deleting Multiple Objects

First select the group of objects to be moved or deleted:

1. Place the cursor on the page where it is not over an object, click and drag. A red rectangle will be drawn to show the area you are selecting.
   When you release the mouse button, all objects inside the red rectangle will be highlighted.
2. You can modify the group selected by holding the control key and clicking on an object. That highlights the object if it was not previously highlighted, unhighlights it otherwise.
   You can skip the click and drag described above and select the whole group with control/clicks.

You'll notice that when more than one object is selected, no property editor is displayed. If you want to edit an object after highlighting a group, click on the object (without holding control). That will unhighlight all other objects.

5.1 Moving

You can now move the selected group by putting the cursor in any of the striped borders, clicking and dragging.

Note: You should check the global settings before moving a group of objects. If `"Snap elements to grid?"` is true, each element is independently moved to the nearest grid point after the move. That may not be what you want unless they are all on grid points before the move.

5.2 Deleting

You can delete all selected objects by clicking on `Delete` in the `Edit` menu.

Note that when a group of objects is removed with `Delete` they are not put into the paste buffer and so cannot be reclaimed with `Paste`.

6 Viewing Live Data

The flow of telemetry data to a graphics page, both when it is being modified in the page builder and when it is later displayed via a STOL `PAGE` directive, is controlled by the `Start`, `Stop`, and `Properties...` items under the `Data` menu.

6.1 Start / Stop Data

In the page builder, data is not sent by default, so the item labelled `"Start"` is enabled. When a page is displayed by other means data is sent by default, so this item is initially disabled.

When the `"Start"` item is clicked, the page tries to get telemetry mnemonics from ITOS. If it is successful, the message `"Receiving data"` is displayed at the top of the page and values should start to appear in objects whose mnemonic or expression properties have been set.

The `"Start"` button now changes to disabled. The `Stop` item should now be enabled and clicking on it should stop the flow of mnemonics to the page and display the message `"User stopped data"` at the top of the page.

If the page is not successful in getting mnemonics from ITOS, it displays the message `"Not connected to the data point server"` and the `"Start"` item remains enabled. That probably means ITOS is not running on your machine. If you know of another machine where ITOS is running, you can specify it via the `"Data Server"` properties dialog described below. If you start ITOS on your machine, click the `Properties...` item and select `"Start Data"`.

When a graphics page is displayed with the STOL PAGE directive, the STOL directives `PAGE FREEZE` and `PAGE THAW` have the same effect as clicking the `"Stop"` and `"Start"` items.

6.2 Data Server Properties Dialog

Clicking on the `Properties...` item in the `Data` menu displays a window for monitoring communications between the page and the data point server of an active ITOS.

The following buttons at the bottom of the window may be useful:

`Clear` Clears the window.

`Ping` Send a ping command to the data point server.

`Start Data / Stop Data`
Starts and stops all data coming from ITOS.
When finished pages are viewed from STOL or via a web browser, pressing `"Stop Data"` here stops data to all pages currently displayed. Pressing the `"Stop"` item in the `Data` menu stops data only to that page.

`Set Host...`
Allows you to specify a running ITOS which will supply telemetry values. It displays a dialog with two input fields, `"Hostname"` and `"Port"`. By default the host is set to `"localhost"`. You can change that to the name or IP address of

a machine where ITOS is running and click "OK". The page will try to contact that ITOS.
There is usually no need to change the port number.

`Close` Closes this data server window.

If a page displays the "Receiving data" message but one or more objects are not displaying data, some things to check are:

1. Look in the Data Server window for messages like "Error from dps: ? XXX is not a mnemonic". That means an object specified a mnemonic named "XXX", which is not defined in the current database.
2. In the page builder, highlight the object and insure that its expression or mnemonic property is set.
3. If the object contains one or more expressions, check the "expression status" field in the object's property editor. If the status is not "OK", there is a syntax error to be corrected.

6.3 Update Rate

Values on graphics pages are changed whenever new telemetry is received. When a stripchart object is created, it is possible to specify how often a global time mnemonic is sent to the page.

When a graphics page is started from STOL with a PAGE directive, an update rate can be specified, as with a text page, but that rate is ignored.

7 Getting information at runtime

You can get information on page objects at runtime by left-clicking on them. For example:

- Clicking on a mnemonic displays the mnemonic's name, type, raw and converted values and any flags set for the mnemonic. If the mnemonic's name is displayed in blue, clicking on it brings up an instance of netscape showing database information about the mnemonic.
- Clicking on an expression displays the expression's text, type and current value. If the expression contains a syntax error, that is displayed. Links are included for all mnemonics in the expression and clicking on one of them gives information on that mnemonic.
- Clicking on a gauge gives information on the expression whose value is displayed in the gauge.
- Clicking on a plot or stripchart gives information on each expression shown in the plot.
- Right-clicking on a command button gives the directive executed when the button is pressed.
  NOTE that you must right-click to get information about a command button. (Left-click executes the command.)

8 Static Text Objects

These objects display text that never changes.

Changable properties of static text objects:

`BackgroundColor`
: Specifies the background used with the text. If you change this color, this background will always be used, even if the page bakcground is changed in the future.

`Font`
: Specifies the text's font.

`ForegroundColor`
: Specifies the color used to display the text. If you change this color, the text will always use the color you select, even if the page foreground is changed in the future.

`Text`
: Specifies the text to be displayed. When you press return after entering the text, the new text is displayed in the object and the object is expanded if necessary. If you decrease the size of the text, the object is not automatically shrunk, so you'll have to do that manually.

9 Mnemonic Objects

These objects are used to display information on a single telemetry mnemonic. It is possible to display any combination of:

- The mnemonic's raw value
- The menmonic's converted value (if a conversion is defined in the database)
- An indication when the mnemonic is static
- The mnemonic's limit status
- An indication when the mnemonic has bad quality

In a single mnemonic object, the fields which are displayed are always displayed in the order listed above. If some other order is needed, you'll have to create two or more mnemonic objects.

Mnemonic objects have the following changable properties:

`Background color`
: Specifies the background used with the text. If you change this color, this background will always be used, even if the page bakcground is changed in the future.

`converted format`
: Gives the format used for displaying the converted value. Formats are given in C-style '%' specifications. Documentation for the STOL functions FORMAT and STRFDATE give details on formats accepted. If the format contains a length specification and the

`ConvertedFormatStatus`
: After you enter a converted format, this field displays any format errors discovered or `"OK"`. This field cannot be changed.

`converted max length`
: Specifies the maximum number of characters used to display the converted value. This value over-rides any length implied by the format specification. This property is ignored if the converted value is not being displayed.

`Default`
: When you click on `"settings"` a default editor is displayed. It contains most of the properties of mnemonics objects. Any settings you make in this editor will be the default settings in future mnemonic objects you create.

`Font`
: These three properties are used just like those in Static Text objects.

`Foreground color`
: Specifies the color used to display the text. If you change this color, the text will always use the color you select, even if the page foreground is changed in the future.

`converted value displayed?`
: Set this to true to display the mnemonic's converted value.

`limit flag displayed?`
: When this property is true, a two-character limit flag is displayed. The flag is blank if the mnemonic is in limits. Otherwise it is one of
  - RL (Red Low)
  - YL (Yellow Low)
  - YH (Yellow High)
  - RH (Red High)
  - UN (Unassigned)
  - QA (Bad Quality)
  - DV (Delta Violation)
  - >> (Rail High)
  - << (Rail Low)

`quality indicator displayed?`
: When this property is true, a one-character quality flag is displayed. If the mnemonic has bad quality, a 'Q' is displayed. Otherwise a blank is displayed.

`raw value displayed?`
: Set this to true to display the mnemonic's raw value.

`static indicator displayed?`
: When this property is true, an asterisk '*' is displayed when the mnemonic is static.

`mnemonic`
: The name of the mnemonic to be displayed. You'll notice that when you press return after entering the mnemonic's name it is changed to uppercase.
  The page builder does not insure that the mnemonic name you enter is defined in any mission database. Therefore it is a good idea to start live data if possible to confirm that the mnemonic is recognized.

`raw format`
: Gives the format used for displaying the raw value. Formats are given in C-style '%' specifications. Documentation for the STOL functions FORMAT and STRFDATE give details on formats accepted. If the format contains a length specification and the

`RawFormatStatus`
: After you enter a raw format, this field displays any format errors discovered or `"OK"`. This field cannot be changed.

  `"RawMaxLength"` property has not been set, the format determines how large the raw field is.
  If no format is entered, a default will be used.
  This property is ignored if the raw value is not being displayed.

`RawMaxLength`
: Specifies the maximum number of characters used to display the raw value. This value over-rides any length implied by the format specification.
  This property is ignored if the raw value is not being displayed.

10 Expression Objects

An expression object can display the value of an arithmetic expression including any number of mnemonics and any STOL functions.

Changeable expression properties:

`BackgroundColor`
: Specifies the background used with the text. If you change this color, this background will always be used, even if the page bakcground is changed in the future.

`ExprFont` Specifies the text's font.

`ExprName` If you enter a name here, other objects can use this name as a mnemonic name. The value of this expression is the raw value of that pseudo mnemonic.
Two things to be cautious about when naming expressions:

1. The name should not duplicate a mnemonic name defined in the database.
2. The name should not duplicate a name defined by another object, even on another page. If the two pages were displayed simultaneously the values of the two expressions would overwrite one another.

`Expression`
: The expression whose value is displayed.

`ExpressionStatus`
: After you enter an expression, this field displays any syntax errors discovered or `"OK"`. This field cannot be changed.

`ForegroundColor`
: Specifies the color used to display the text. If you change this color, the text will always use the color you select, even if the page foreground is changed in the future.

`Format` Gives the format used for displaying the expression. Formats are given in C-style '%' specifications. Documentation for the STOL functions FORMAT and STRFDATE give details on formats accepted.
If no format is entered, a default will be used.

`FormatStatus`
: After you enter a format, this field displays any format errors discovered or `"OK"`. This field cannot be changed.

11 Bar Gauge Objects

Bar gauges are oriented vertically by default. To make one horizontal, change its "IsVertical" property to false and resize it to be wider than it is high.

A gauge can display the value of an arithmetic expression including any number of mnemonics and any STOL functions.

Changable bar gauge properties:

`BackgroundColor`
: Specifies the background used with the bar gauge. If you change this color, this background will always be used, even if the page bakcground is changed in the future.

`Bar Color` Sets the color of the bar in the gauge.

`Expression`
: The expression whose value is displayed in the gauge.

`ExpressionStatus`
: After you enter an expression, this field displays any syntax errors discovered or "OK". This field cannot be changed.

`Font` Specifies the font used by the text in the bar gauge. The font used in a bar gauge is not affected by the font setting in the global properties.

`ForegroundColor`
: Specifies the color used by the outline, text labels, and tick marks of the bar gauge.

`HashDelta`
: This property is ignored unless "HashPlacementIsDefault", the previous property, is false. When that property is false, hash marks are placed "HashDelta" units apart.
If the value entered causes an absurd number of hash marks to be printed, an error is printed and the value is ignored.

`Hash Placement Is Default`
: When true, the gauge decides where hash marks are placed. When false, the value of "HashDelta", the next property, determines where hash marks are placed.

`IsVertical`
: When true, the gauge's scale and bar are vertical. When false they are horizontal.

`LabelDelta`
: This property is ignored unless "LabelPlacementIsDefault", the previous property, is false. When that property is false, the first label printed is "MinValue". The next label is at `MinValue + LabelDelta` and so on until `MaxValue` is reached.
If the value entered causes an absurd number of labels to be printed, an error is printed and the value is ignored.

`Label Placement Is Default`
: When true, the gauge decides which labels are printed. When false, the value of `"LabelDelta"`, the next property, determines which labels are printed.

`MaxValue`
: The value at the top of the gauge's scale. A value lower than `"MinValue"`, the previous property, will be ignored.

`MinValue`
: The value at the bottom of the gauge's scale. A value larger than `"MaxValue"`, the next property, will be ignored.

12 Dial Gauge Objects

A dial gauge is a circular gauge with a needle that can display the value of an arithmetic expression including any number of mnemonics and any STOL functions. You can set the gauge's shape from a full circle to a small arc. You can also set whether the gauge increases clockwise or counter clockwise.

Changable dial gauge properties:

`BackgroundColor`
: Specifies the color used in the rectangular part of the object outside the dial. If you don't set this property, the page's background color will be used.

`DialColor`
: Specifies the color used for the dial background not affected by `"Red high"` etc.

`Position of max`

`Position of min`

`Gauge increases clockwise?`
: These three properties specify the shape of the gauge and which direction the needle moves. The positions of `"min"` and `"max"` are given in degrees from the 6:00 position. Positive degrees are measured clockwise, negative degrees are measured counter clockwise.
Notice the default values. `"Position of min"` is +90, meaning 90 degrees clockwise from 6:00, or 9:00 and that's where the minimum value of the gauge is placed. `"Position of max"` is -90, meaning 90 degrees counter clockwise from 6:00, or 3:00.
`"Gauge increases clockwise"` is true by default, so the gauge dial goes clockwise from 9:00 to 3:00, passing through 12:00. If `"Gauge increases clockwise"` is changed to false, the dial will increase counter clockwise, passing through 6:00.
To construct a gauge that appears to cover 360 degrees, it is necessary to set `"Position of min"` and `"Position of max"` to slightly different values, such as 1 and -1.

`Expression`
: The expression whose value is displayed in the gauge. It can include any number of mnemonics and any STOL functions.

`ExpressionStatus`
: After you enter an expression, this field displays any syntax errors or `"OK"`. This field cannot be changed.

`Font` Specifies the font used in labels.

`HashDelta`
: Specifies where hash marks are placed. There is always a mark at `"minimum value"`. The next mark is at `"minimum value"` plus `"HashDelta"` and so on around the gauge. Gauges are usually clearer when `"HashDelta"` is the same as `"LabelDelta"`.

`LabelDelta`
: Specifies which labels are printed. There is always a label at "`minimum value`". The next label is at "`minimum value`" plus "`LabelDelta`" and so on around the gauge.

`Maximum Value`
: The highest value on the gauge's scale. A value lower than "`Minimum Value`", the previous property, will be ignored.

`Minimum Value`
: The lowest value on the gauge's scale. A value larger than "`Maximum Value`", the next property, will be ignored.

`Red high value`

`Yellow high value`

`Yellow low value`

`Red low value`
: These values put a red or orange background on part of the gauge dial. (Orange is used instead of yellow because it is difficult to find pointer and text colors that contrast with both red and yellow.)
The dial is colored red from "`Red high value`" to "`Maximum value`". The dial is colored orange from "`Yellow high value`" to "`Red high value`" or "`Maximum value`", whichever is lower. The low values work similarly.
The value "`None`" can be entered to remove a background color.

`NeedleColor`
: Specifies the color used for the gauge's needle.

`TextColor`
: Specifies the color used for labels, hash marks and the dial borders.

13 Command Button Objects

A command button executes a single STOL directive when pressed.

Changable plot properties are:

`Background color`
: The button's background color.

`Button text`
: Specifies the text displayed on the button.

`Command` Specifies the STOL directive to be executed.

`Font` The font used on the button.

`Foreground color`
: The button's foreground color.

When the button's page is started by ITOS, the command is immediately executed when the button is pressed. When the button's page is started by a web browser or by an application like selectpage, most commands are not executed. The one exception is that PAGE directives which display a single page are executed.

14 Plot Objects

A plot object can graph one to four expressions including any number of mnemonics and any STOL functions. Expressions are always plotted as functions of time. The time mnemonic to be used is specified by changing the "Time mnemonic" property described below.

Plot objects are quite flexible, but do not handle high data rates well. Stripcharts are better at plotting data that arrives one or more times a second.

The Y axis of these plots can scale to match the data displayed, or you can set constant minimum and maximum values for the Y axis.

When a plot object is displaying data, you can zoom to any area of the plot by right-clicking and dragging to define the zoom rectangle. You can scroll horizontally or vertically by holding a shift key while right-clicking and dragging.
After zooming or scrolling, return to normal display by clicking on the plot and typing "/r".

For zooms, scrolls or "/r" to work, the plot must have focus. In the page builder that may require two clicks because the first click brings up the plot's property editor and gives it focus.

Changable plot properties are:

`BackgroundColor`
: Specifies the plot's background color.

`ForegroundColor`
: Specifies the color used for axes and gridlines.

`Graph1`

`Graph2`

`Graph3`

`Graph4`
: Each of these properties specifies one graph on the plot. There is no difference between the four graphs except their default colors. Clicking on any of them brings up a graph editor, described below.
It may take a few seconds for the graph editor to appear after you click on "No expression".

`More properties`
: Clicking on "properties" displays the "Plot More Editor" described below. It allows you to edit global plot properties.

`X Axis Expression`
: Specifies which mnemonic or STOL expression that supplies X values for data points.

`X Expression status`
: Shows the result of parsing the current expression. That is either "OK" or a syntax error.

The **Graph Editor** displayed when you select one of the "Graph" properties has the following fields:

`Expression`
: Specifies the expression to be plotted.

`Expression status`
: Shows the result of parsing the current expression. That is either `"OK"` or a syntax error.

`Line color`
: Allows you to change the color of the line used to display this graph.

`Data point shape`
: Allows you to change the shape of data points on this graph.

The **Plot More Editor** is displayed when you select the `"More properties"` property. It has the following fields:

`Header` Specifies text to be displayed at the top of the plot.

`Footer` Specifies text to be displayed at the bottom of the plot.

`Maximum number of points`
: Specifies the maximum number of points that will be displayed on any graph. When that number of data points is reached, the oldest point is discarded.

`Y axis logarithmic?`
: Can display data on a logarithmic scale.

`Legend visible?`
: When true, a legend is displayed which shows the expression plotted by each of the graphs along with its symbol and line color.

`Legend location`
: Specifies where the legend is displayed. Ignored unless `"Display a legend"` is true.

`Horizontal gridline spacing`
: The default value `"No lines"` means that no horizontal gridlines are displayed. To display gridlines, replace `"No lines"` with the distance you want between lines.

  To remove gridlines, enter `"No lines"` or simply `"N"` in this field.

  If the value you enter in this field is too small, a dialog box will be displayed with the message `"Provided grid spacing is too small to show up"`. You may occasionally get that warning even if the spacing is appropriate for the data, but you have not entered Y axis min and max values as described below. Entering those min and max values will stop the warnings. If you want gridlines and also want the Y axis to autoscale, you can choose to ignore the warnings. They will not be displayed when the page is used in ITOS.

`Vertical gridline spacing`
: The default value `"No lines"` means that no vertical gridlines are displayed. To display gridlines, replace `"No lines"` with the number of seconds you want between lines. Fractions of a second are acceptable.
  To remove vertical gridlines, enter `"No lines"` or simply `"N"` in this field.

`Y axis min`

`Y axis max`

The default value `"Autoscale"` in these fields means the Y axis will be autoscaled to fit the data displayed. You can replace that string with constant values for the min and max values on the Y axis.

To return to autoscaling after entering a value, enter `"Autoscale"` or simply `"A"` in these fields.

A much more extensive property editor can be brought up by middle clicking on a plot. This editor is available in the page builder and also when the page is displayed. A complete description of this editor is beyond the scope of this manual, but one useful fact is that input fields sometimes do not recognize the backspace key. Delete should be used instead.

15 Stripchart objects

Stripcharts are similar to plots. They can graph one to four expressions including any number of mnemonics and any STOL functions. Expressions are always plotted as functions of time and the time mnemonic used is specified in the property editor.

Stripcharts are optimized to handle high data rates, but are less flexible than plots in several ways. They cannot zoom or scroll and the Y axis is fixed. They do not accept expressions as X values, but always plot Y expressions against a single time mnemonic.

Changable stripchart properties are:

`BackgroundColor`
: Specifies the plot's background color.

`ForegroundColor`
: Specifies the color used for axes and gridlines.

`Graph1`

`Graph2`

`Graph3`

`Graph4`
: Each of these properties specifies one graph on the stripchart. There is no difference between the four graphs except their default colors. Clicking on any of them brings up a graph editor, described below.
It may take a few seconds for the graph editor to appear after you click on `"No expression"`.

  The **Graph Editor** displayed when you select one of the `"Graph"` properties has the following fields:

  `Expression`
  : Specifies the expression to be plotted.

  `Expression status`
  : Shows the result of parsing the current expression. That is either `"OK"` or a syntax error.

  `Line color`
  : Allows you to change the color of the line used to display this graph.

`LegendShowing`
: When this property is set to true, the expressions being plotted are listed at the bottom of the chart along with the color used for each expression.

`SlotDelta`

`SlotWidth`
: A stripchart is divided into vertical "slots", each `"SlotWidth"` pixels wide. Each data point is one slot wide. Adjacent slots represent times `"SlotDelta"` seconds apart, so `"SlotDelta"` should be set to the expected time difference between data points. Or it can be set to a larger value to spread out data on the chart.

Note that slotdelta need not be changed for playback as long as the time mnemonic used is a telemetry mnemonic rather than a global like GBL_GMTOFF. During a playback, telemetry mnemonics arrive more frequently than during realtime operation, but their values still differ by the same amounts.

`TimeMnemonic`
: Specifies the mnemonic that will supply X values for the chart. The chart ignores `"P@"` entered with this mnemonic. The converted value will automatically be used if it is available.

`YHashDefault`

`YHashDelta`
: If `"YHashDefault"` is true, the stripchart will decide where hash marks are drawn on the Y axes and ignore `"YHashDelta"`. Otherwise hash marks are drawn every `"YHashDelta"` units.

`YLabelDelta`

`YLabelsDefault`
: If `"YLabelsDefault"` is true, the stripchart will decide which labels are displayed on the Y axes and ignore `"YLabelDelta"`. Otherwise labels are displayed every `"YLabelDelta"` units.

`YMax`

`YMin` Specify the minimum and maximum values on the Y axis.

16 Rectangle Objects

These objects draw a rectangular shape on the page.

Changable properties of rectangle objects:

`Background color`
: Specifies the background of the component, see the '`Background filled`' property for more information.

`Content area filled`
: Specifies whether or not the interior of the component is filled. If this is `true` then the foreground color will be used to fill the component, if this is `false` then the interior of the component will be transparent, allowing the components that are drawn behind this component to show through.

`Foreground color`
: Specifies the foreground of the component, see the '`Content area filled`' property for more information.

`Background filled`
: Specifies the opacity of the component's background. If this is `true` then the background color will be used to fill the background of the component, if this is `false` then the background will be transparent, allowing the components that are drawn behind this component to show through.

`Stroke color`
: Specifies the color of the outline surrounding the component, see the '`Stroke drawn`' property for more information.

`Stroke drawn`
: Specifies whether or not the outline surrounding the component is drawn. If this is `true` then the stroke color will be used to outline the component, if this is `false` then the outline of the component will not be drawn.

`Stroke width`
: The width of the outline surrounding the component in pixels.

17 RoundRectangle Objects

These objects draw a rectangular shape with rounded corners on the page.

Changable properties of round rectangle objects:

`Arc height`
: Specifies the height of the arced corners in pixels.

`Arc width`
: Specifies the width of the arced corners in pixels.

`Background color`
: Specifies the background of the component, see the '`Background filled`' property for more information.

`Content area filled`
: Specifies whether or not the interior of the component is filled. If this is `true` then the foreground color will be used to fill the component, if this is `false` then the interior of the component will be transparent, allowing the components that are drawn behind this component to show through.

`Foreground color`
: Specifies the foreground of the component, see the '`Content area filled`' property for more information.

`Background filled`
: Specifies the opacity of the component's background. If this is `true` then the background color will be used to fill the background of the component, if this is `false` then the background will be transparent, allowing the components that are drawn behind this component to show through.

`Stroke color`
: Specifies the color of the outline surrounding the component, see the '`Stroke drawn`' property for more information.

`Stroke drawn`
: Specifies whether or not the outline surrounding the component is drawn. If this is `true` then the stroke color will be used to outline the component, if this is `false` then the outline of the component will not be drawn.

`Stroke width`
: The width of the outline surrounding the component in pixels.

18 Ellipse Objects

These objects draw an elliptical shape on the page.

Changable properties of ellipse objects:

`Background color`
: Specifies the background of the component, see the '`Background filled`' property for more information.

`Content area filled`
: Specifies whether or not the interior of the component is filled. If this is `true` then the foreground color will be used to fill the component, if this is `false` then the interior of the component will be transparent, allowing the components that are drawn behind this component to show through.

`Foreground color`
: Specifies the foreground of the component, see the '`Content area filled`' property for more information.

`Background filled`
: Specifies the opacity of the component's background. If this is `true` then the background color will be used to fill the background of the component, if this is `false` then the background will be transparent, allowing the components that are drawn behind this component to show through.

`Stroke color`
: Specifies the color of the outline surrounding the component, see the '`Stroke drawn`' property for more information.

`Stroke drawn`
: Specifies whether or not the outline surrounding the component is drawn. If this is `true` then the stroke color will be used to outline the component, if this is `false` then the outline of the component will not be drawn.

`Stroke width`
: The width of the outline surrounding the component in pixels.

19 Line Objects

These objects draw a line on the page.

Changable properties of line objects:

`Anchor 1` Specify the first anchor point of the line segment. Valid locations are '`NORTH_WEST`', '`SOUTH_WEST`', '`NORTH_EAST`', and '`SOUTH_EAST`'.

`Anchor 2` Specify the second anchor point of the line segment. Valid locations are '`NORTH_WEST`', '`SOUTH_WEST`', '`NORTH_EAST`', and '`SOUTH_EAST`'.

`Background color`
: Specifies the background of the component, see the '`Background filled`' property for more information.

`Background filled`
: Specifies the opacity of the component's background. If this is `true` then the background color will be used to fill the background of the component, if this is `false` then the background will be transparent, allowing the components that are drawn behind this component to show through.

`Stroke color`
: Specifies the color of the line, see the '`Stroke drawn`' property for more information.

`Stroke drawn`
: Specifies whether or not the line is drawn. If this is `true` then the stroke color will be used to draw the line, if this is `false` then the line will not be drawn.

`Stroke width`
: The width of the line in pixels.

20 Arc Objects

These objects draw an arc shape on the page.

Changable properties of arc objects:

`Background color`
: Specifies the background of the component, see the '`Background filled`' property for more information.

`Content area filled`
: Specifies whether or not the interior of the component is filled. If this is `true` then the foreground color will be used to fill the component, if this is `false` then the interior of the component will be transparent, allowing the components that are drawn behind this component to show through.

`Angular extent`
: Specifies the number or degrees that the arc makes from the '`Start Angle`' location. Negative values are clockwise, positive values are counter-clockwise.

`Foreground color`
: Specifies the foreground of the component, see the '`Content area filled`' property for more information.

`Background filled`
: Specifies the opacity of the component's background. If this is `true` then the background color will be used to fill the background of the component, if this is `false` then the background will be transparent, allowing the components that are drawn behind this component to show through.

`Start Angle`
: Specifies the starting location of the arc in degrees from three o'clock. Negative values move the starting angle clockwise, positive values move the starting angle counter-clockwise.

`Stroke color`
: Specifies the color of the outline surrounding the component, see the '`Stroke drawn`' property for more information.

`Stroke drawn`
: Specifies whether or not the outline surrounding the component is drawn. If this is `true` then the stroke color will be used to outline the component, if this is `false` then the outline of the component will not be drawn.

`Stroke width`
: The width of the outline surrounding the component in pixels.

`Closure type`
: Specifies the way in which the arc shape is closed. '`OPEN`' specifies that no line segment connects the ends of the arc. '`CHORD`' specifies that the arc is closed by a line segment that goes from the start point to the ending point. '`PIE`' specifies that the arc is closed by two line segments that go from the endpoints to the center of the full ellipse created by the arc.

21 Label Objects

These objects draw a label on the page.

Changable properties of label objects:

`Start angle`
: Specifies the starting location of the text in degrees from three o'clock. Negative values move the starting angle clockwise, positive values move the starting angle counter-clockwise.

`Background color`
: Specifies the background of the component, see the '`Background filled`' property for more information.

`Font`
: Specifies the font used to render the text in the label.

`Foreground color`
: Specifies the foreground of the component, see the '`Content area filled`' property for more information.

`Background filled`
: Specifies the opacity of the component's background. If this is `true` then the background color will be used to fill the background of the component, if this is `false` then the background will be transparent, allowing the components that are drawn behind this component to show through.

`Text`
: Specifies the text to be displayed in the label. The string '`Label`' is displayed if no text has been specified.

22 Makepage

makepage is a Bourne shell script that starts the page builder, allowing a user to create or modify graphics telemetry pages. It can be run from a shell prompt or by clicking on the STOL `"prog"` button and selecting `"Display builder"`.

It accepts the following command line arguments:

`-version` Prints the page builder's version number and exits.

23 selectpage

Selectpage is a Bourne shell script that resides in '`/usr/tcw/bin`'. It provides an alternate way to view telemetry pages.

It might be used to view telemetry from an ITOS running on another host on the local network.

Selectpage accepts two command line arguments:

`-host <name>`
: Specifies the name or IP address of the host where ITOS is running. The default is localhost.

`-cgidir <path>`
: This argument is ignored when displaying graphics pages. It is used for text-based pages and its default value is `/home/${ITOS_MISSION}/public_html/classes`. It specifies a directory containing dir_list.cgi, page_parser.cgi and select_page.cgi. That directory must have a sibling directory or link named `"pages"` where the page tree is rooted.

To view text-based telemetry pages, `".page"` files, it is necessary for ITOS to be running on the local host. That is not necessary for graphics pages.

24 Page Object Requirements

This section describes requirements for ITOS beans (other than the usual bean requirements like an argumentless constructor and get.. and set.. methods for properties)

- Any class that will appear in the bean box's toolbox list and will be visible on pages must implement the VisibleBean interface. A comment in VisibleBean.java explains the single method in this interface.
- All beans that get data from the data point server get it from a single instance of class DataSource. An object that wants to get data from DataSource must:
  1. Implement the DataConsumer interface. Its methods are described in DataSource.java.
  2. Get a pointer to the current DataSource with a call like:
     ```
     dataSource = PageGlobal.getDataSource(PageGlobal.CREATE);
     ```
  3. Tell the DataSource which mnemonics the object wants to receive with code like:
     ```
     String[] list = new String[n];
     list[0] = mneName;
        :
     dataSource.exprNames(this, list);
     ```
  4. Define `setVariable` (part of the DataConsumer interface) so that it returns quickly. In particular, `setVariable` must not make any graphics calls. If it does, the DataSource hangs while display manager tasks are performed. `setVariable` usually just saves its data and wakes another thread to deal with it.
- We want pages to continue to work after bean class files change. To make that happen:
  1. All bean classes must include a data member like:
     ```
     static final long serialVersionUID = 1L;
     ```
     The serialVersionUID value can be treated like a version number. If the serialized form of an object changes in such a way that it is incompatible with the previous form, the serialVersionUID may be changed to prevent Java from attempting to deserialize objects that were stored in the previous form.
  2. All beans that implement the Externalizable interface must provide an implementation for both the `readExternal()` and the `writeExternal()` methods.

  See `readExternal()` and `writeExternal()` in Shape.java for examples.
- Some serialization issues that are not obvious:

  When using the default Java serialization mechanism, a bean's base class is always serialized before the bean. To remedy that issue, the java.io.Externalizable interface should be implemented instead of the java.io.Serializable interface. Externalizable objects have full control over their own serialized form.

  An example of the problem with the default Java serialization mechanism is as follows:
  1. ItosPlot extends JCChart. That means JCChart is serialized before ItosPlot.
  2. ItosPlot extends Container and has a JCChart member. ItosPlot must add the JCChart object, so JCChart is serialized along with the Container base class.

- It is very nice when a user can give a bean focus by clicking on the bean, rather than being forced to click on the invisible Wrapper around the bean. We get that for free unless one of these is true:
  1. The bean contains subcomponents.
  2. The bean itself uses mouse clicks.

  MneBean is an example of the first condition. It is a Panel which adds Label objects that completely cover the Panel. To allow mouse clicks to get to the Wrapper, MneBean does the following:
  - Implements MouseListener.
  - Registers itself as listener to mouse events on the Labels.
  - Its MouseListener methods simply pass the mouse event to the Wrapper. It would be nice to simply register the Wrapper as the MouseListener, but MneBean's constructor doesn't have a pointer to the Wrapper.

  If a bean needs to accept all clicks, there's nothing we can do. But if it needs clicks on only one or two buttons, clicks on the remaining buttons could be passed to the Wrapper.
- BeanInfo files can specify the order in which properties are displayed in the property editor by calling

  ```
  setValue("propertyNumber", n)
  ```

  where n is an Integer giving an editor row number. The first row is 1. The string "propertyNumber" is defined in ItosBeanDefaults.propertyNumber. The following example puts the editor for YHashDelta in row 12:

  ```
  PropertyDescriptor pd12 = new PropertyDescriptor(
              "YHashDelta", FPBean.class);
  pd12.setValue(ItosBeanDefault.propertyNumber, new Integer(12));
  ```
- It is a good idea to define public Dimension getMinimumSize(). The default method seems to return a Component's current size and as a result it is not possible to shrink the bean.
- The Convert applications require that all of a bean's properties be classes for which toString() is defined. We get this for free with most properties. Some exceptions are:
  - MneBean Default property, whose type is MneBean. As a result, MneBean defines toString().
  - ItosPlot properties Graph1, ..., Graph4, whose types are GraphInfo. Therefore GraphInfo defines toString().
  - FPBean properties Graph1, ..., Graph4, whose types are FPGraphInfo. Therefore FPGraphInfo defines toString().
- Most beans, when outside the beanbox, respond to user clicks by displaying information about the mnemonics or expressions they contain. The ShowInfo methods do most of the work, but the beans must implement MouseListener.
- Finally, the names of all bean classes that are to appear in the '`Tool Box`' must be put into the '`src/rsrc/ITOS/javadisplay/beans/ItosBeans.list`' file. The '`make`

`install`' process will include that file in the '`itos-all.jar`' file where the '`ITOS Page Builder`' reads it when it starts up.